An Answer to a Conjecture on Overlaps in Partial Words Using Periodicity Algorithms*

F. Blanchet-Sadri[1], Robert Mercaş[2], Abraham Rashin[3], and Elara Willett[4]

[1] Department of Computer Science, University of North Carolina, P.O. Box 26170, Greensboro, NC 27402–6170, USA, blanchet@uncg.edu
[2] GRLMC, Universitat Rovira i Virgili, Plaça Imperial Tárraco, 1, Tarragona, 43005, Spain, robertmercas@gmail.com
[3] Department of Mathematics, Rutgers University, 110 Frelinghuysen Rd., Piscataway, NJ 08854–8019, USA
[4] Department of Mathematics, Oberlin College, 10 North Professor St., King 205, Oberlin, OH 44074-1019, USA

Abstract. We propose an algorithm that given as input a full word w of length n, and positive integers p and d, outputs (if any exists) a maximal p-periodic partial word contained in w with the property that no two holes are within distance d. Our algorithm runs in $O(nd)$ time and is used for the study of freeness of partial words. Furthermore, we construct an infinite word over a five-letter alphabet that is overlap-free even after the insertion of an arbitrary number of holes, answering affirmatively a conjecture from Blanchet-Sadri, Mercaş, and Scott.

1 Introduction

In [1], Manea and Mercaş extend the concept of repetition-freeness of full words to partial words which are sequences over a finite alphabet that may contain some "do not know" symbols called "holes." There, several problems regarding cube-freeness are investigated. Following the same lines, in [2], Blanchet-Sadri, Mercaş and Scott consider the concepts of square- and overlap-freeness. In these papers, the authors investigate the existence of infinite full words into which arbitrarily many holes can be inserted without introducing repetitions (inserting a hole is defined as replacing a letter with a hole in a fixed position of a word, the length of the word remaining the same). A constraint that no two holes can be placed one or two positions apart is needed, to avoid trivial squares and cubes. This problem is equivalent to determining whether an infinite partial word w can be found such that, none of its factors of length kp, for any positive integer p and

* This material is based upon work supported by the National Science Foundation under Grant No. DMS–0754154. We thank the referees of a preliminary version of this paper for their very valuable comments and suggestions. This work was done during the second author's stay at the University of North Carolina at Greensboro. A World Wide Web server interface has been established at www.uncg.edu/cmp/research/freeness2 for automated use of the program.

rational $k \geq 2$, is 2-valid and p-periodic. A partial word is called d-valid if any positions i, j satisfying $0 < |i - j| \leq d$ are not both holes.

A well-known result of Thue states that over a binary alphabet there exist infinitely many overlap-free infinite full words [3, 4]. In [1], the question was raised as to whether there exist overlap-free infinite partial words, and to construct them over a binary alphabet if such exist. In [2] and [5], the authors settle this question by showing that over a two-letter alphabet there exist overlap-free infinite partial words with one hole, and none exists with more than one hole. Moreover, in [2] it is shown that there exist infinitely many overlap-free infinite partial words with an arbitrary number of holes over a three-letter alphabet. There, it is also proved that there exists an infinite overlap-free word over a six-letter alphabet that remains overlap-free after an arbitrary selection of its letters are changed to holes, and none exists over a four-letter alphabet. The case of a five-letter alphabet remained open.

Conjecture 1 ([2]). There exists an infinite word over a five-letter alphabet that remains overlap-free after an arbitrary 2-valid insertion of holes.

In this paper, we investigate the question of which finite full words can be made periodic by insertion of holes, with the restriction that no two holes be too close together. More precisely, we present an algorithm that determines whether a finite full word w of length n contains a d-valid p-periodic partial word, in $O(nd)$ time. As a consequence, we give a positive answer to Conjecture 1. An overview of basic concepts of combinatorics on partial words follows.

Let A be a nonempty finite set of symbols called an *alphabet*. Each element $a \in A$ is called a *letter*. A *full word* over A is a finite sequence of letters from A. A *partial word* over A is a finite sequence of letters from $A_\diamond = A \cup \{\diamond\}$, the alphabet A extended with the hole symbol $\diamond$ (a full word is a partial word that does not contain the $\diamond$ symbol). A partial word u of length n over A can be viewed as a function $u : \{0, \ldots, n-1\} \to A_\diamond$. The *length* of a partial word u is denoted by $|u|$ and represents the total number of symbols in u. The *empty word* is the sequence of length zero and is denoted by ε. The powers of a partial word u are defined recursively by $u^0 = \varepsilon$ and for $n \geq 1$, $u^n = uu^{n-1}$. The set containing all finite full words over the alphabet A is denoted by A^*, while the set of all finite partial words over the alphabet A is denoted by $A^*_\diamond$.

A *strong period* of a partial word u over A is a positive integer p such that $u(i) = u(j)$ whenever $u(i), u(j) \in A$ and $i \equiv j \ mod \ p$. In such a case, we say u is *strongly p-periodic*. A *weak period* of u is a positive integer p such that $u(i) = u(i + p)$ whenever $u(i), u(i + p) \in A$. In such a case, we say u is *weakly p-periodic*. The word $abba\diamond bacb$ is weakly 3-periodic, but not strongly 3-periodic.

If u and v are two partial words of equal length, then u is said to be *contained* in v, denoted $u \subset v$, if $u(i) = v(i)$, for all $u(i) \in A$. Partial words u and v are *compatible*, denoted by $u \uparrow v$, if there exists w such that $u \subset w$ and $v \subset w$. A partial word u is a *factor* of a partial word v if $v = xuy$ for some x, y. The factor u is *proper* if $u \neq \varepsilon$ and $u \neq v$. We say that u is a *prefix* of v if $x = \varepsilon$ and a *suffix* of v if $y = \varepsilon$. A full word u is said to contain an overlap if it

Fig. 1. The words w and u with columns 2 and 5 highlighted

contains a factor *avava* (two overlapping occurrences of the word *ava*) with a a letter and v a word [6]. In [1], this definition was extended to partial words, an overlap being considered a factor $a_0v_0a_1v_1a_2$ with $v_0 \uparrow v_1$, and a_0, a_1, a_2 pairwise compatible letters ($a_0v_0a_1v_1a_2 \subset avava$, for some letter a and word v), and it can be generalized as follows: A partial word $a_0v_0a_1v_1a_2$, where $v_0 \uparrow v_1$, is a *weak overlap* if $a_0 \uparrow a_1$ and $a_1 \uparrow a_2$ (because $a_0v_0a_1 \uparrow a_1v_1a_2$), and a *strong overlap* if a_0, a_1, a_2 are pairwise compatible symbols. Note that a strong overlap is also a weak overlap. A partial word is *weakly overlap-free* (respectively, *strongly overlap-free*) if none of its factors is a weak (respectively, strong) overlap.

2 Periodic Partial Words With No Two Holes Within a Fixed Distance

We say two positions i, j in a partial word u are *d-proximal* if $0 < |j - i| \leq d$, where d denotes a positive integer. We say that u obeys the *hole constraint* d (or u is *d-valid*) if no two holes in u are d-proximal. When the value of d is clear from context, we may suppress reference to it, simply referring to the "hole constraint" or to "proximal" positions.

Let w be a length n full word defined over an alphabet A of size k. In this section, we present an $O(nd)$ time algorithm, which finds, for given positive integers d and p both less than n, a *d-valid* p-periodic partial word contained in w, if any exists. In other words, it determines whether it is possible to insert holes into w with no two holes within distance d, such that the resulting partial word has strong period p. If this is possible, such a word is returned.

In order to work with words of length n more easily, we write them in rows of length p. For a partial word u and for an integer x, $0 \leq x < p$, we will call *column* x the sequence of positions (or letters at these positions) $x, x + p, \ldots, x + lp$, where l is the maximal integer such that $x + lp < n$. For example in Figure 1, if $w = abadecabbdeeaba$, $p = 6$, and $d = 2$, then $u = abadecab\diamond de\diamond aba$ is obtained using our algorithm. For an integer x, $0 \leq x < p$, let $S_x = \{w(i) \mid 0 \leq i < n, i \equiv x \bmod p\}$ be the set of distinct letters appearing in column x of w. We construct a new set of partial words $\Omega = \{\omega \mid \omega(i) \in S_i\}$, and call $u \subset w$ a partial word *induced* by the choice $\omega \in S_0 \times S_1 \times \cdots \times S_{p-1}$, if $u \subset \omega^l$, for some rational l. Now, u, induced by ω, is d-valid if and only if for any two proximal positions i and j ($0 \leq i, j < n$, $0 < |i - j| \leq d$), it is not the case that $u(i) = \diamond = u(j)$.

Remark 1. The choice of letters $\omega \in \prod_{x=0}^{p-1} S_x$ induces a d-valid word if and only if for every two proximal positions i, j, $u(i) = \omega(i \bmod p)$ or $u(j) = \omega(j \bmod p)$.

This suggests a geometric approach for determining which choices of letters do not cause a hole constraint violation. For $(a_0, b_0) \in A^2$, let the *cross centered at* (a_0, b_0) be the set $+(a_0, b_0) = \{(a, b) \in A^2 \mid a = a_0 \text{ or } b = b_0\}$. Then, the choices of letters a for column x and b for column y that do not cause any hole constraint violations, are precisely those in the intersection of the crosses centered at $(w(i), w(j))$, for i, j proximal positions in columns x, y.

The subsets of A^2 formed by intersecting crosses, however, are of special forms. The following theorem describes these forms, and shows that they can be determined in l-linear time, where l is the number of crosses that need to be intersected (the number of distinct ordered pairs (i, j) where i, j are proximal positions in columns x, y, respectively).

Theorem 1. *Considered as a set of entries in a $k \times k$ matrix, any set T formed by intersecting crosses must be either: (1) FULL: A^2; (2) CROSS(a_0, b_0): a cross $+(a_0, b_0)$; (3) ROW(a_0): a row of the matrix $\{(a_0, b) \mid b \in A\}$; (4) COL$(b_0)$: a column of the matrix $\{(a, b_0) \mid a \in A\}$; (5) TWO$((a_1, b_1), (a_2, b_2))$: a set of two points (a_1, b_1) and (a_2, b_2) in neither the same row nor column; (6) ONE(a_0, b_0): a singleton set $\{(a_0, b_0)\}$; or (7) NULL: the null set $\emptyset$.*

Proof. Let m be the number of crosses that are intersected: $T = \bigcap_{s=1}^{m} +(a_s, b_s)$. If $m = 0$, then $T = A^2$ is FULL. The form FULL is only possible for $m = 0$. If $m = 1$, then $T = +(a_1, b_1)$ is CROSS(a_1, b_1). Now suppose that $m > 1$ and let $T' = \bigcap_{s=1}^{m-1} +(a_s, b_s)$. We consider what happens when we intersect $+(a_m, b_m)$ with T', for T' in each of the above forms.

Let $T' = \text{CROSS}(a_0, b_0)$. If $(a_m, b_m) = (a_0, b_0)$, then $T' = +(a_m, b_m)$, so $T = T'$. If $a_m = a_0$ and $b_m \neq b_0$, then $T = \text{ROW}(a_0)$. If $b_m = b_0$ and $a_m \neq a_0$, then $T = \text{COL}(b_0)$. If $a_m \neq a_0$ and $b_m \neq b_0$, then $T = \text{TWO}((a_0, b_m), (a_m, b_0))$. Therefore, intersecting $+(a_m, b_m)$ with a CROSS matrix results in a CROSS, ROW, COL, or TWO matrix, as depicted in Figure 2. a). If $T' = \text{ROW}(a_0)$ and $a_m = a_0$, then $T = T' \subset +(a_m, b_m)$. Otherwise, $T = \text{ONE}(a_0, b_m)$. If $T' = \text{COL}(b_0)$ and $b_m = b_0$, then $T = T' \subset +(a_m, b_m)$. Otherwise, $T = \text{ONE}(a_m, b_0)$.

Now, let $T' = \text{TWO}((a, b), (a', b'))$. If (a_m, b_m) is equal to (a, b') or to (a', b), then $T = T' \subset +(a_m, b_m)$. Now, if $a = a_m$ or $b = b_m$ then $(a, b) \in +(a_m, b_m)$, but $(a', b') \notin +(a_m, b_m)$, and so $T = \text{ONE}(a, b)$. Similarly, if $a' = a_m$ or $b' = b_m$ then $T = \text{ONE}(a', b')$. Finally, if $a \neq a_m$, $b \neq b_m$, $a' \neq a_m$ and $b' \neq b_m$, then $(a, b), (a', b') \notin +(a_m, b_m)$, so $T = \text{NULL}$. Therefore, intersecting $+(a_m, b_m)$ with a TWO matrix results in a TWO, ONE, or NULL matrix, as depicted in Figure 2. b). If $T' = \text{ONE}(a_0, b_0)$ and $a_m = a_0$ or $b_m = b_0$, then $T' \subset +(a_m, b_m)$, so $T = T'$. Otherwise, $T = \text{NULL}$. Finally, if $T' = \text{NULL}$, then $T = \text{NULL}$. □

Now, returning to the question of which $\omega \in \prod_{x=0}^{p-1} S_x$ induce d-valid partial words, for two columns $x, y < p$, we define the constraint matrix M^{xy}, to be a $k \times k$ matrix such that, for all $a, b \in A$, $M^{xy}(a, b)$ is $*$ if for every pair of proximal positions i, j in columns x, y, $(a, b) \in +(w(i), w(j))$, and 0 otherwise. Note that, trivially, the constraint matrix from x to y is the transpose of the constraint matrix from y to x, and that $\omega \in \prod_{x=0}^{p-1} S_x$ induces a d-valid partial word if and only if for every $x, y \in \{0, \ldots, p-1\}$, $M^{xy}(\omega(x), \omega(y)) = *$.

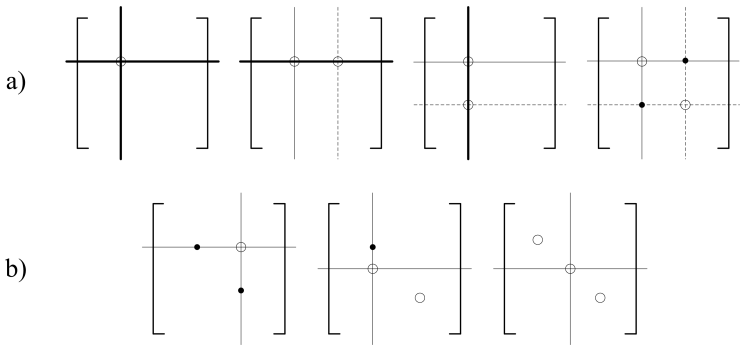

Fig. 2. Intersection of different matrices

The result of Theorem 1 is that the constraint matrices can be classified into a few simple types. Therefore, in practice, we store constraint matrices as objects that encode the form of the matrix (FULL, CROSS, TWO, etc.), and at most four characters to denote rows and columns (querying the position of stars in row a_0 of the object $<$ TWO$, (a, b), (a', b') >$ yields b if $a_0 = a$, b' if $a_0 = a'$ and NONE otherwise). These can be constructed and read in constant time.

Remark 2. If columns x, y are proximal, that is x, y contain proximal positions, then $0 < |x - y| \leq d$ or $0 < p - |x - y| \leq d$.

Fix some variables that will be shared by the algorithms: a table of constraint matrices, M; sets F_{ROW}, F_{ONE}, F_{TWO} and F_{CROSS}, where F_{FORM} contains (x, y) for which M^{xy} is of form FORM; a list of letters ω, where $\omega(x)$ is the letter chosen for column x. The following lemmas will be useful in proving the validity of our algorithms.

Lemma 1. *If $0 \leq x, y < p$ with $0 < |x - y| \leq d$, then M^{xy} is not FULL.*

Proof. The positions x and y in w are proximal since $0 < |x - y| \leq d$. Therefore at least one cross (namely, that centered at $(w(x), w(y))$ is used in the creation of the matrix M^{xy}, so it cannot be FULL. □

Furthermore, it follows from Theorem 1 that the types of constraints that one column can exert on another are limited.

Lemma 2. *If two columns x, y with $0 \leq x < y < p$, contain each at least two different letters, and M^{xy} is a CROSS matrix, then $|x - y| \geq \max\{p - d, d + 1\}$.*

Proof. Since M^{xy} is not a FULL matrix, by Remark 2, we have that $|x - y| \leq d$ or that $p - d \leq |x - y|$. Suppose that $|x - y| \leq d$, and let $y + sp$ be a position in column y, where $y \leq y + sp < n$. Thus, $x + sp$ is a position in column x, since $0 \leq x \leq x + sp < y + sp < n$. Furthermore, every position in column y is proximal to some position in column x. Since M^{xy} is a CROSS matrix, all

Algorithm 1 Initalizing the matrices

```
for (x, y) columns within d do
    M^{xy} := FULL
for i = 0 to n − y step p do
    intersect M^{xy} with cross centered at (w(x + ip), w(y + ip))
```

ordered pairs $(w(i), w(j))$, for i, j proximal positions in columns x, y, must be equal. Therefore all letters in column y of w are equal, a contradiction. Therefore $|x - y| > d$ and $|x - y| \geq p - d$, so $|x - y| \geq \max\{p - d, d + 1\}$. □

There exist even more restrictions regarding CROSS matrices.

Lemma 3. *Let x_1, x_2, x_3 be distinct columns with at least two different letters each. If $M^{x_2x_3}$ and $M^{x_1x_3}$ are CROSS matrices, then $M^{x_1x_2}$ is neither a FULL nor a CROSS matrix.*

Henceforth, by *columns within d* we mean columns x, y such that $0 < |x-y| \leq d$ or $0 < p - |x - y| \leq d$. Any other pair of columns is necessarily related by a FULL constraint matrix and therefore can be ignored. Algorithm 1 computes all non-FULL constraint matrices of w in $O(nd)$ time.

Corollary 1. *The forms (as per Theorem 1) of all the non-FULL constraint matrices for w can be determined in $O(nd)$ time via Algorithm 1.*

Note that given two proximal columns x and y, and a letter a chosen for column x, there are either zero, one, or $\|S_y\|$ choices of a letter for column y that do not conflict with the choice of letter a for column x. This observation suggests an algorithm for labeling multiple columns. Let us now construct a directed graph G that has vertex set $\{0, \ldots, p-1\}$ and edge set consisting of edges (x, y) labelled by M^{xy} when columns x, y are within d.

Theorem 2. *For a column x and a letter $a \in S_x$, Algorithm 2 correctly chooses letters for some additional columns such that, after the completion of this algorithm no undetermined column is constrained by an already determined column. Additionally, if the constraint matrices have already been computed, the running-time of Algorithm 2 is $O(m)$, where m is the number of edges that are traversed.*

Proof. The problem of finding a choice of letters for the columns is equivalent to finding a labeling of the vertices of G, such that every vertex x is labeled with a letter $\omega(x)$ that occurs in column x of w, and for any two columns x and y within d, the $(\omega(x), \omega(y))$-entry of M^{xy} is a $*$. If such a labeling exists, then it induces a p-periodic d-valid partial word contained in w, by replacing every non-$\omega(x)$ letter in each column x with a hole.

The algorithm starts by assuming a labeling of vertex x by the letter a, and then performs a breadth-first search on the graph G, starting at x. This is implemented using a queue. Suppose that a vertex y has been marked by letter b and that we are now traversing an edge from y to z. Then the constraint matrix

Algorithm 2 Fill(x, a)

```
 1: initialize Q to be an empty queue accepting columns
 2: choose letter a for column x
 3: add x to Q
 4: while dequeue y from Q do
 5:     let b = ω(y)
 6:     for z a neighbor of y do
 7:         let row be the b row of the matrix M^{yz}
 8:         remove edges between y and z
 9:         if row has all ∗'s then
10:             next (go to line 4)
11:         else if row has exactly one ∗, say at position c then
12:             if letter c has already been chosen for column z then
13:                 next (go to line 4)
14:             else if column z is unlabeled then
15:                 choose letter c for column z
16:                 add z to Q
17:                 next (go to line 4)
18:         undo all recent labelings and edge erasures
19:         return false
20: return true
```

M^{yz} either uniquely determines the label c on the sink vertex z, or it imposes no constraint at all, or there are no choices, in which case the algorithm immediately fails. In the former case, either the unique label is applied ($\omega(z)$ is set to c) and vertex z is added to the queue for later traversal, or if $\omega(z)$ has already been set to a different value, the algorithm fails because there cannot be any labeling of G with $\omega(x) = a$ and $\omega(z)$ with its original value. In the case when no constraint is imposed (the b row is filled with $*$'s), this matrix is ignored, since for any value of $\omega(z)$ the matrix will not cause a contradiction. In all cases, the edge (y, z) and its opposite (z, y) are marked as having been traversed, so that they will not be visited again. In conclusion, an undetermined column is marked exactly when it is constrained by an already determined column, thus, ensuring that at the end of the algorithm no determined column will constrain an undetermined column. This algorithm visits m edges, no more than once each. On each edge, it performs a constant time operation. Thus, Algorithm 2 runs in $O(m)$ time.

Please note that undoing all recent labelings and edge erasures, while keeping the algorithm's runtime within $O(m)$, is solved in constant time by implementing data structures that could be "marked" in a particular state, and reset to this state later on. These data structures are used for the sets of neighbors of a vertex, the sets F_{FORM} (of edges of each type), and the set of labeled vertices. While, all the F_{FORM}'s and labelings can be reset in constant time, the vertex neighbor sets can be reset in $O(l)$ time, where l is the number of vertices visited during this run of the algorithm. Since the number of vertices visited is less than the number of edges visited, $l < m$, the overall algorithm runs in $O(m)$ time. □

Algorithm 3 Traversing the entire graph

initialize matrices
for (x, y) columns within d **do**
    if $M^{xy} = \text{NULL}$ **then**
        return false
    add (x, y) to F_{FORM}
for column x **do**
    if $\|S_x\| = 1$ **then**
        Fill$(x, w(x))$
while exists (x, y) with M^{xy} of form ROW(a), in F_{ROW} **do**
    if not Fill(x, a) then return false
while exists (x, y) with M^{xy} of form ONE(a, b), in F_{ONE} **do**
    if not Fill(x, a) then return false
while exists (x, y) with M^{xy} of form TWO$((a, b), (a', b'))$, in F_{TWO} **do**
    if not Fill(x, a) and not Fill(x, a') then return false
for column x **do**
    if column x is unlabeled **then**
        choose $w(x)$ for column x
for i from 0 to $n - 1$ **do**
    let $u(i)$ be $w(i)$ if $w(i) = \omega(i \bmod p)$ and $\diamond$ otherwise
return u

The next lemma will help us prove that we never need to run Algorithm 2 ("Fill(x, a)") on a vertex more than twice.

Lemma 4. *Suppose that x and y are vertices of G such that $M^{xy} = TWO((a, b), (a', b'))$, Fill$(x, a)$ returns true, and $\omega \in \prod_{z=0}^{p-1} S_z$ induces a d-valid partial word u with $\omega(x) = a'$. Then, there exists a choice ω' of letters for the columns, that induces a d-valid partial word with $\omega'(x) = a$.*

Proof. Let T be the set of vertices of G that are labeled by Fill(x, a), and Q be the labeling of T. For every vertex x of G, let $\omega'(x) = Q(x)$ if $x \in T$ and $\omega'(x) = \omega(x)$ otherwise. Since the labeling Q of T was generated by Fill(x, a), we know that no letter choice for a vertex outside T is constrained by any of the letter choices specified in Q. Furthermore, since ω induced a d-valid partial word, we know that no constraint matrix is violated by two letter choices in ω. Therefore the letter choices in ω' do not violate any constraint matrices, so ω' induces a d-valid partial word. Also, clearly $\omega'(x) = a$, so we have our result. □

The next algorithm traverses all edges corresponding to non-FULL matrices and finds a consistent labeling of the vertices of G if any exists.

Theorem 3. *Algorithm 3 returns a d-valid p-periodic partial word contained in w, unless no such word exists. The running-time of the algorithm is $O(nd)$.*

Proof. If there is a NULL matrix between two columns, then no consistent labeling of the vertices exists, so the algorithm fails. If any column in w has all letters equal, then that letter must be assigned for the column, and Fill$(x, w(x))$

ran. There can only be one consistent labeling of all vertices if it succeeds (note that the determination of whether a column has only one character can be performed in $O(\frac{n}{p})$ time, and thus, it can be performed for all columns in $O(n)$ time). Similarly, if there is a ROW or ONE matrix M^{xy} with a $*$ in row a, then a must be chosen for column x. We run Fill(x, a), and it must succeed for there to be a consistent labeling of the vertices of G.

If $M^{xy} = \text{TWO}((a, b), (a', b'))$ then we know that any consistent labeling of the vertices of G must have column x labeled with either a or a'. But by Lemma 4, if some consistent labeling of G exists and Fill(x, a) returns true, then there exists a consistent labeling of G that agrees on all choices of letters made by Fill(x, a). Therefore in this case we can simply continue. Otherwise we try Fill(x, a'). If this fails, then we return false.

At this point in the algorithm, any unlabeled vertices x, y are related by either a FULL or CROSS matrix, since all other types of matrices have already been taken into account. Consider a graph T' with the so-far unlabeled vertices of G as the vertex set, and an edge between x and y if and only if M^{xy} is a CROSS matrix. We can satisfy all remaining constraints (the CROSS matrices) by considering every connected component of T' separately. But, by Lemma 3, this graph has no connected components of size greater than two (since only crosses are left, connecting more than two of them falls in Lemma 3).

We claim that we can label any remaining vertex x with $w(x)$ (the first letter appearing in column x) without introducing any new contradictions. This is clearly true for any isolated vertex in T', since these are unconstrained. Now consider x, y vertices in T' related by CROSS(a, b). Every proximal pair of positions i, j in columns x, y must have $w(i) = a$ and $w(j) = b$. But between any two columns that have proximal pairs, at least one of them has its first (top) position proximal to some position in the other column. Therefore $w(x) = a$ or $w(y) = b$ (or both). Therefore these choices satisfy the constraint matrix. If the algorithm reached this step, then there exists a p-periodic d-valid partial word contained in w, namely the one induced by ω.

Each matrix is visited at most twice (this worst case scenario is achieved precisely if the edge is examined twice in the loop starting on line 13). There are at most $2pd$ matrices in question, and analyzing a row of a matrix takes constant time. Thus, the running-time is $O(pd)$ plus the running-time of checking which columns are uniform, and of constructing the constraint matrices ($O(nd)$ by Corollary 1). Therefore, the total running-time of Algorithm 3 is $O(nd)$. □

3 Short Factors in the Images of Morphisms

Let $\beta : A^* \to A^*$ be a non-erasing prolongable morphism on $z_0 \in A$. For $m \geq 0$, let $z_{m+1} = \beta(z_m)$, and $w = \lim_{m \to \infty} z_m$ the fixed point of β. Let $F_m(y)$ denote the set of length m factors of y. Since z_m is a proper prefix of z_{m+1}, for any $n \geq 1$:

$$F_n(z_0) \subseteq F_n(z_1) \subseteq F_n(z_2) \subseteq \cdots \subseteq \bigcup_{m \geq 0} F_n(z_m) = F_n(w)$$

But since $F_n(w)$ is finite, having at most $|A|^n$ elements, using the pigeonhole principle, the chain must become constant after finitely many steps.

Lemma 5. *Let $m \geq 0, n \geq 1$ be such that $\emptyset \neq F_n(z_m) = F_n(z_{m+1})$. Then $F_n(z_m) = F_n(w)$.*

Suppose now, that $q = \min\{|\beta(a)| \mid a \in A\} \geq 2$ and $F_2(z_m) = F_2(w)$. In other words, β maps every letter of the alphabet to a word of length at least two, and all length two factors of w are factors of z_m. For all $i \geq 0$, it can be shown that $F_{q^i+1}(z_{m+i}) = F_{q^i+1}(w)$. Moreover, let $s = \max\{|\beta(a)| \mid a \in A\}$. Then it can also be shown that the set of length n factors of w can be computed in $O(n^{\log_q s})$ time. Hence, for any $n \geq 2$, $z_{m+\lceil \log_q(n-1)\rceil}$ has all length n factors of w, and this set can be computed in polynomial time. Furthermore, if β is a uniform morphism, we have $q = s$ and $F_n(w)$ is computable in $O(n)$ time. Note that in some cases we can discard the requirement $q \geq 2$, by taking a higher iteration of the morphism (for $\mu : a \mapsto abc$, $b \mapsto ac$, $c \mapsto b$, the square $\mu^2 : a \mapsto abcacb$, $b \mapsto abcb$, $c \mapsto ac$, can be used in the above theorems, since it generates the same fixed point).

4 An Overlap-Free Word Over an Alphabet of Size Five

Note that the definition of weak overlap generalizes the overlap definitions used in [2] and [5], since here a factor is considered to be an overlap of length $2p+1$ if it has p as a weak period, while in [2, 5], the factor had to have a strong period p. In this section, we generate an infinite full word over a 5-letter alphabet, which remains weakly overlap-free after any 2-valid insertion of holes. First, define a morphism $\gamma : \{a, b, c, d\}^* \to \{a, b, c, d\}^*$ with $\gamma(a) = ad$, $\gamma(b) = ac$, $\gamma(c) = cb$, and $\gamma(d) = ca$. Since a is a prefix of $\gamma(a)$, γ is prolongable. Thus, we define the fixed point of γ, $\Gamma = \lim_{i\to\infty} \gamma^i(a)$. Consider some properties of Γ.

Remark 3. Both $\gamma^3(a) = adcacbad$ and $\gamma^4(a) = adcacbadcbacadca$ have only ac, ad, ba, ca, cb and dc as their length two factors. Thus, by Lemma 5, these are the only length two factors of Γ.

Lemma 6. *The infinite full word Γ is square-free.*

Proof. It suffices to show that every $\gamma^n(a)$ is square-free. Clearly $\gamma^0(a) = \varepsilon$ is square-free. Now let $n \geq 0$ and assume that $\gamma^n(a)$ is square-free. Suppose, for contradiction, that $\gamma^{n+1}(a)$ has a square factor of length $2p$ starting at position i. Since the letters b and d appear only at odd positions of $\gamma^{n+1}(a)$, hence, if p is odd, the factor would be in $\{a, c\}^*$. Since all binary words of length 4 contain squares, it must be that $p = 1$, which is a contradiction according to Remark 3 ($p = 1$ in order to avoid the contradiction of having squares).

Therefore p must be even. If i is even, since $\gamma^{n+1}(a) = \gamma(\gamma^n(a))$ it follows that $\gamma^n(a)$ contains a square, contradicting the initial assumption. Hence, i is odd. Since, $\gamma(f)$ ends in a different letter for all $f \in \{a, b, c, d\}$, it follows that we have a factor that is a square starting at position $i - 1$, which is an even position. Following the previous reasoning we again reach a contradiction. □

Now let $\delta : \{a, b, c, d\}^* \to \{f, g, h, i, j\}^*$ be a morphism defined by $\delta(a) = fgifh$, $\delta(b) = fghij$, $\delta(c) = jigjh$, and $\delta(d) = jihgf$. We claim that $\delta(\Gamma)$ is overlap-free after an arbitrary (2-valid) insertion of holes.

Proposition 1. *There are no factors of $\delta(\Gamma)$ of length ≤ 21 that can be turned into weak overlaps by any 2-valid insertion of holes.*

Proof. Using a variant of Algorithm 3, we checked that for $p \leq 10$, there is no factor of $\delta(\Gamma)$ of length $2p+1$ that contains a 2-valid weakly-p-periodic word. □

Recall the following result from [2].

Remark 4. [2] Full words $t = t_0t_1t_2$ and $s = s_0s_1s_2$ contain compatible 2-valid partial words if and only if for some i, $t_i = s_i$.

Lemma 7. *In $\delta(\Gamma)$, any two length seven sequences starting with the same character will contain at least three consecutive mismatches if they are not identical.*

Proof. According to Remark 3 the only length two factors of Γ are ac, ad, ba, ca, cb and dc. We prove the lemma for sequences starting with letter f, the other cases being similar. If a sequence starts with f, then it must be either $fgifhji$, a prefix of both $\delta(ac)$ and $\delta(ad)$, $fghijfg$, prefix of $\delta(ba)$, $fjigjhf$, first factor starting with f in both $\delta(dca)$ and $\delta(dcb)$, or, $fhjigjh$ and $fhjihgf$, suffixes of $\delta(ac)$ and $\delta(ad)$. It is easy to check that each two of these blocks contain three consecutive mismatches once aligned. □

Proposition 2. *No factor of $\delta(\Gamma)$ of length $2p + 1 > 21$ with p not divisible by 5 can be turned into a weak overlap by a 2-valid insertion of holes.*

Proof. Assume that there exists $a_0v_0a_1v_1a_2$, a factor that can be transformed into a weak overlap after insertion of holes, where each v_i is a word of length $p-1$ and the a_j's are letters. Since p is not divisible by 5, it follows that the images of δ in a_0v_0 and a_1v_1 will not be aligned. If the second letters of a_0v_0 and a_1v_1 are equal, then we get a contradiction by Lemma 7 (here no two corresponding length seven subwords in each half starting with the same character can be identical). If the two positions do not match, following Remark 4 it must be that either the first or the third positions must match. Using the same technique we get a contradiction in both these cases. Therefore, no factor of $\delta(w)$ of length $2p + 1 > 21$ with p not divisible by 5 can be turned into a weak overlap. □

Proposition 3. *No factor of $\delta(\Gamma)$ of length $2p + 1 > 21$ with p divisible by 5 can be turned into a weak overlap.*

Proof. Assume towards a contradiction that a factor $a_0v_0a_1v_1a_2$ can be transformed into a weak overlap after insertion of holes. Since $|a_0v_0| = 5k$, for some $k > 2$, it follows that the images of δ will be aligned in a_0v_0 and a_1v_1. By looking at the blocks of δ we see that only the images of b and d do not contain three consecutive mismatches once aligned. Hence, we will consider the case when these two images are aligned, the other cases being straightforward by Remark 4.

Note that the only character preceding d in Γ is a, and the only character preceding b is c, while the only character following d in Γ is c, and the only character following b is a. Assume that the block determined by $\delta(d)$ ends before the last position in $a_i v_i$ with $i \in \{0, 1\}$. The character following this block is j, while the one following the block $\delta(b)$ is f. Note that this letter together with the last two characters of the block gives us the sequences gfj and ijf, that will not match after a valid insertion of holes, by Remark 4.

If the block $\delta(d)$ starts at a position greater than 5, it follows that it is preceded by $\delta(a)$. Since $\delta(a)$ will align with a block $\delta(c)$ according to the previous observations, by Remark 4 we conclude that a matching is impossible. □

Theorem 4. *The infinite word $\delta(\Gamma)$ over a 5-letter alphabet is weakly overlap-free after an arbitrary insertion of holes.*

Proof. This follows directly from Propositions 1, 2, and 3. □

Since strong periodicity implies weak periodicity, the theorem answers an open problem of [2] regarding how large an alphabet must be to create an infinite word that is strongly overlap-free despite arbitrary insertions of holes.

Corollary 2. *The infinite word $\delta(\Gamma)$ over a 5-letter alphabet is strongly overlap-free after an arbitrary insertion of holes.*

Please note that the lower bound of five letters presented in [2] stands, since for alphabets of size smaller than five, all infinite words contain factors of length $2p + 1$ that are strongly p-periodic, and therefore weakly p-periodic. Also note that the use of the terms weakly and strongly overlap-free word comes from the concepts of weak- and strong-periodicity (when looking at overlaps from this point of view, the terminology comes naturally).

References

1. Manea, F., Mercaş, R.: Freeness of partial words. Theoretical Computer Science **389** (2007) 265–277
2. Blanchet-Sadri, F., Mercaş, R., Scott, G.: A generalization of Thue freeness for partial words. Theoretical Computer Science **doi:10.1016/j.tcs.2008.11.006** (2008)
3. Thue, A.: Uber unendliche Zeichenreihen. Norske Vid. Selsk. Skr. I, Mat. Nat. Kl. Christiana **7** (1906) 1–22
4. Thue, A.: Uber die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. Norske Vid. Selsk. Skr. I, Mat. Nat. Kl. Christiana **1** (1912) 1–67
5. Halava, V., Harju, T., Kärki, T.: Overlap-freeness in infinite partial words. Technical Report 888, Turku Centre for Computer Science (2008)
6. Lothaire, M.: Combinatorics on Words. Cambridge University Press (1997)